# **📋 TitleTesterPro OAuth Implementation Plan**

## **🎯 Goal:**

✅ Users log in with Google & connect their YouTube channel(s).  
✅ Works for verified app & verified scopes.  
✅ Handles expired/missing tokens.  
✅ Supports multi-channel users (brand accounts).  
✅ Supports incremental consent flow.  
✅ Includes fallback UI & instructions if tokens are invalid.

# **🪜 Step-by-Step Plan**

## **1️⃣ 📄 Scopes & OAuth Consent**

### **Scopes Required:**

✅ Phase 1:

ts

CopyEdit

['openid', 'email', 'profile']

✅ Phase 2:

ts

CopyEdit

[

'https://www.googleapis.com/auth/youtube.readonly',

'https://www.googleapis.com/auth/youtube',

'https://www.googleapis.com/auth/youtube.force-ssl',

'https://www.googleapis.com/auth/yt-analytics.readonly'

]

Add all 7 to Google Cloud → OAuth Consent Screen → Scopes.  
✅ Save & publish.  
✅ Add your test emails in the **Test Users** section.

Ensure app is already verified for sensitive scopes (as you said it is).

## **2️⃣ 🌐 Environment Variables**

In .env:

bash

CopyEdit

NEXT\_PUBLIC\_SUPABASE\_URL=

NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY=

SUPABASE\_SERVICE\_ROLE\_KEY=

GOOGLE\_CLIENT\_ID=

GOOGLE\_CLIENT\_SECRET=

GOOGLE\_REDIRECT\_URI=https://yourapp.com/api/auth/callback

ENCRYPTION\_KEY=<32\_char\_random\_string>

## **3️⃣ 🗃 Database Changes**

### **Add columns (if missing) to accounts table:**

sql

CopyEdit

ALTER TABLE accounts

ADD COLUMN IF NOT EXISTS access\_token TEXT,

ADD COLUMN IF NOT EXISTS refresh\_token TEXT,

ADD COLUMN IF NOT EXISTS youtube\_channel\_id TEXT,

ADD COLUMN IF NOT EXISTS youtube\_channel\_title TEXT,

ADD COLUMN IF NOT EXISTS expires\_at BIGINT;

### **Create temp\_oauth table:**

sql

CopyEdit

CREATE TABLE IF NOT EXISTS temp\_oauth (

user\_id TEXT PRIMARY KEY,

access\_token TEXT NOT NULL,

refresh\_token TEXT NOT NULL,

expires\_at BIGINT NOT NULL,

channels JSONB NOT NULL,

created\_at TIMESTAMP DEFAULT NOW()

);

## **4️⃣ 🧑‍💻 Authentication Flow**

### **Phase 1 — Google Login**

* Button triggers Supabase .signInWithOAuth with:

ts

CopyEdit

provider: 'google',

options: {

scopes: ['openid', 'email', 'profile'].join(' ')

}

* On success → user session created & basic info saved.

### **Phase 2 — YouTube Connection**

* After login, check if accounts.youtube\_channel\_id is populated.
* If not → prompt:

“Connect your YouTube Account to start testing your titles.”

* Button triggers .signInWithOAuth with:

ts

CopyEdit

provider: 'google',

options: {

scopes: [

'https://www.googleapis.com/auth/youtube.readonly',

'https://www.googleapis.com/auth/youtube',

'https://www.googleapis.com/auth/youtube.force-ssl',

'https://www.googleapis.com/auth/yt-analytics.readonly'

].join(' '),

queryParams: {

access\_type: 'offline',

prompt: 'consent'

}

}

## **5️⃣ 📄 Backend Components**

### **Files to Implement:**

✅ /api/auth/callback.ts → Handles OAuth code, exchanges for tokens, fetches channels, saves to DB.  
✅ /pages/select-channel.tsx → UI for multi-channel users to pick channel.  
✅ /api/save-channel.ts → Saves selected channel to DB & cleans up temp\_oauth.  
✅ /lib/utils/refreshAccessToken.ts → Refreshes token if expired.  
✅ /lib/db.ts → Handles upsertAccount, saveTempTokens, getTempTokens, deleteTempTokens.

## **6️⃣ 🔐 Encryption**

Tokens must be encrypted before saving:

ts

CopyEdit

import crypto from 'crypto';

export function encryptToken(token: string): string {

const iv = crypto.randomBytes(16);

const key = crypto.createHash('sha256').update(process.env.ENCRYPTION\_KEY!).digest();

const cipher = crypto.createCipheriv('aes-256-cbc', key, iv);

const encrypted = Buffer.concat([cipher.update(token, 'utf8'), cipher.final()]);

return iv.toString('hex') + ':' + encrypted.toString('hex');

}

export function decryptToken(data: string): string {

const [ivHex, encryptedHex] = data.split(':');

const iv = Buffer.from(ivHex, 'hex');

const encrypted = Buffer.from(encryptedHex, 'hex');

const key = crypto.createHash('sha256').update(process.env.ENCRYPTION\_KEY!).digest();

const decipher = crypto.createDecipheriv('aes-256-cbc', key, iv);

const decrypted = Buffer.concat([decipher.update(encrypted), decipher.final()]);

return decrypted.toString('utf8');

}

## **7️⃣ 🔄 Token Refresh**

Implement automatic refresh using /lib/utils/refreshAccessToken.ts.  
Called whenever YouTube API responds 401 or when expires\_at is near.

## **8️⃣ 💬 Fallback UX**

### **When OAuth Fails:**

✅ Show a branded error page:

html

CopyEdit

<div>

<h1>😕 YouTube Connection Error</h1>

<p>Something went wrong while connecting your YouTube account.</p>

<p>This may happen if permissions expired or were revoked.</p>

<ol>

<li>Go to <a href="https://myaccount.google.com/permissions">Google Account Permissions</a></li>

<li>Remove <strong>TitleTesterPro</strong></li>

<li>Come back & log in again.</li>

</ol>

<a href="/login"><button>Reconnect</button></a>

</div>

## **9️⃣ 🧪 Testing Checklist**

✅ Login → Dashboard loads basic user info.  
✅ YouTube not connected → prompted to connect.  
✅ Multiple channels → select one → saved.  
✅ Tokens expire → automatically refreshed.  
✅ Invalid tokens → error page with revoke instructions.  
✅ DB accounts table populated with encrypted tokens & channel info.  
✅ DB temp\_oauth table cleared after channel is selected.

## **🔗 What the Agent Needs to Deliver**

✅ All the above files written & placed in the proper folders.  
✅ .env configured correctly.  
✅ Google Cloud OAuth scopes updated & saved.  
✅ DB migrations executed.  
✅ Frontend buttons & prompts updated for Phase 1 + Phase 2 flows.  
✅ Fallback error page implemented.  
✅ Logs in place to debug token errors.  
✅ SQL queries to clear invalid tokens (optional).

## **📜 Optional: SQL to Clear Broken Tokens**

sql

CopyEdit

DELETE FROM accounts WHERE user\_id = '<your-user-id>';

DELETE FROM temp\_oauth WHERE user\_id = '<your-user-id>';

## **🔗 Recap Deliverables:**

✅ Incremental consent flow (Phase 1 + Phase 2).  
✅ All code files: callback, select-channel page, save-channel API, refresh logic, encryption utils, db helpers.  
✅ Frontend prompts & fallback UX.  
✅ SQL migrations & cleanup script.  
✅ Logs & test cases.